Санкт-Петербургский государственный университет

Фундаментальная информатика и информационные технологии
Информационные технологии

Искрич Дмитрий Павлович

Разработка и реализация генетического алгоритма для генерации правил торговой системы на основе анализа исторических данных

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент кафедры информатики
Санкт-Петербургского государственного
университета Григорьев Д. А.

Рецензент:
д.т.н., профессор, декан факультета
информационных технологий и управления
Санкт-Петербургского государственного
технологического института Мусаев А.А.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Informatics and Information Technology
Information Technology

Dmitry Iskrich

Development and implementation of genetic algorithm for generating rules of a trading system based on historical data analysis

Bachelor's Thesis

Scientific supervisor:
Ph. D. of Sc., associate professor Dmitry Grigoriev

Reviewer:
Sc.D., dean of the faculty of Information Technology and Management, St. Petersburg State Technological Institute Musaev Aleksandr Azerovich

Saint-Petersburg
2017

Оглавление

Введение

Одна из центральных задач любого трейдера - подбор прибыльных стратегий для торговли финансовым инструментом. Торговая стратегия трейдера – это формализованные правила для входа и выхода из позиции. Существует множество подходов для создания подобного рода правил, в финансовой индустрии их принято разделять на два больших класса: фундаментальный и технический анализ.

В первом случае, основой для прогнозирования рыночной стоимости инструмента служит обработка производственных показателей компании. Как правило, эти значения демонстрируют состояние дел держателя котировки той или иной акции. Данный подход требует информации, на составление которой необходимы данные за несколько месяцев, а иногда и кварталов. При этом не всегда возможно нахождение взаимосвязей между фундаментальными показателями и рыночной стоимостью, что в свою очередь требует применения исследований, затрагивающих внутренние и глубинные причины изменения валютных курсов. Но даже в этом случае можно придти к причинно-следственным связям, которые будут противоречить друг другу. По различным оценкам лишь 10-20% трейдеров используют фундаментальный анализ, и большинство из них владеет им поверхностно.

В основном торговые правила строятся на методах технического анализа и обработке таких данных, как: текущая цена инструмента, объём торгов, максимальная/минимальная цена за определенный период, цена в момент открытия/закрытия торговой сессии (пример подобного рода данных представлен на рис. 1). На основе серий этих данных строятся так называемые технические индикаторы. В соответствии с постулатами технического анализа технические индикаторы позволяют предсказывать вероятное направление тренда в той или иной бумаге. Таким образом комбинация из индикатора и его значения может служить базисом для торговой стратегии.

Обнаружение прибыльных паттернов является нетривиальной задачей. В данной работе описан метод автоматического построения правил

Date	Open	High	Low	Close	Adj Close*	Volume
Apr 28, 2017	144.09	144.30	143.27	143.65	143.65	20,763,500
Apr 27, 2017	143.92	144.16	143.31	143.79	143.79	14,246,300
Apr 26, 2017	144.47	144.60	143.38	143.68	143.68	20,041,200
Apr 25, 2017	143.91	144.90	143.87	144.53	144.53	18,871,500
Apr 24, 2017	143.50	143.95	143.18	143.64	143.64	17,134,300
Apr 21, 2017	142.44	142.68	141.85	142.27	142.27	17,320,900
Apr 20, 2017	141.22	142.92	141.16	142.44	142.44	23,319,600
Apr 19, 2017	141.88	142.00	140.45	140.68	140.68	17,328,400
Apr 18, 2017	141.41	142.04	141.11	141.20	141.20	14,697,500
Apr 17, 2017	141.48	141.88	140.87	141.83	141.83	16,582,100
Apr 13, 2017	141.91	142.38	141.05	141.05	141.05	17,822,900
Apr 12, 2017	141.60	142.15	141.01	141.80	141.80	20,350,000
Apr 11, 2017	142.94	143.35	140.06	141.63	141.63	30,379,400

Рис. 1: Пример торговых исторических данных, взято с портала Yahoo Finance.

торговой системы без участия человека. Очевидно, что его применение позволит обнаружить большее число эффективных закономерностей поведения цены.

1. Постановка задачи

Целью данной работы является создание системы для генерации торговых правил на базе генетического алгоритма. Для достижения поставленной цели были выделены следующие задачи.

1. Представление торговой стратегии в виде дерева решений.
2. Разработка генетического алгоритма для деревьев решения.
3. Тестирование стратегий на исторических данных.
4. Анализ полученных результатов, составление системных отчётов.

2. Обзор существующих решений

В данной отрасли большое применение имеют системы на базе искусственного интеллекта. С помощью определенного эвристического алгоритма производится поиск паттернов для сигналов покупки и продажи. Как упоминалось ранее, системы отталкиваются от исторических данных за определенный период. Наиболее часто используются данные с периодом в один день. Как правило, трейдер программирует вручную свою торговую систему, а затем оптимизирует ее ключевые параметры методом перебора или эвристическими алгоритмами. Известно большое количество работ, цель которых оптимизация параметров торговых стратегий. Редкое исключение составляют системы генерации правил, которые могут предоставить готовые стратегии, исходя из исторических данных. Для решения данной задачи зачастую наблюдается применение нейронных сетей. В работе [1] данным методом рыночный сигнал классифицируется как *rise* или *fall*, тем самым прогнозируя движения стоимости инструмента. Также имеют место системы на базе мультиагентного подхода, концепция предложенная в системе [2], сконцентрирована на множестве трейдеров-агентов, имеющих некий стартовый капитал и набор правил для торговли. Впоследствии, путём генетического алгоритма система генерирует самого "успешного" агента, чьи правила становятся основой для конечной стратегии.

Конкретно в этой задаче естественной структорой данных является дерево решений.

2.1. Системы на базе деревьев решений

Деревья могут быть применены для интерпретации одного торгового правила. Так, в работе [3] с помощью данной структуры представляются арифметические выражения, которые в свою очередь интрепретируются как паттерн для покупки инструмента. Нередко используется представление всей торговой стратегии в виде дерева решений. На нетерминальных узлах дерева могут располагаться предикаты определяющие переход, а на терминальных (листовых) узлах сигналы к ры-

ночному действию.

Проблема большинства систем заключается в том, что правила привязаны к конкретным вещественным значениям. В данной работе демонстрируется примнение деревьев решения, предикаты которых связаны лишь с относительными уровнями.

2.2. Классификация исторических данных

Нахождение оптимального бинарного дерева решений является NP-полной задачей [4]. Это стало причиной использования различных эвристических алгоритмов, однако применение классических алгоритмов для решения этой задачи (C4.5, ID3) не всегда однозначно в финансовой области.

Существуют системы, основанные на разметке исторических данных указаниями к торговому действию. Этот этап необходим для тренировки некоторых классификаторов, суть которых нахождение взаимосвязи между определенным классом и значениями технических индикаторов. Однако на данный момент отсутствует оптимальный способ такой разметки.

Так, например, в работе [5] для расстановки BUY, SELL и HOLD значений для определенного дня, использовались торговые показатели двух последующих дней. Затем, с использованием ID3 алгоритма на этих данных создавалось дерево решений. В то же время, согласно утверждениям [6] для определения торгового действия может быть недостаточно нескольких последующих значений, важную роль играет общий тренд изменения цены, временная продолжительность которого не всегда очевидна. В системе [7] продемонстрирована расстановка BUY и SELL значений на основе падения или подъёма цены в течение 2 лет на 10 или 15 процентов соответственно. Данные с вычисленными метками используются алгоритмом C4.5 для создания оптимального дерева решений. Этот же алгоритм используется и в работе [8], однако разметка данных для обучения основана на реальных действиях наиболее успешного трейдера в конкретный день.

Таким образом, обилие всевозможных вариантов разметки исторических данных на торговые действия, привело к решению использовать генетический алгоритм в качестве классификатора. Данный подход требует от данных лишь наличия целевой функции для дерева решений, тем самым избавляя от необходимости субъективно размечать исторические данные.

2.3. Системы на базе генетического алгоритма

Генетический алгоритм имеет широкое применение в совокупности с другими эвристиками. В упомянутой ранее системе [2] генетический алгоритм работает на базе мультиагентной концепции. В работе [9] торговая система работает в 2 этапа: в начале, с помощью алгоритма C4.5 создаётся оптимальное дерево решений, затем с использованием генетического алгоритма оптимизируются параметры этой стратегии.

Пример применения генетического программирования для генерации торговых стратегий продемонстрирован в системе *EDDIE* [10]. Используя предложенные пользователем показатели, система тренирует наиболее приспособленное дерево решений. Общая схема работа продемонстрирована на рис. 2

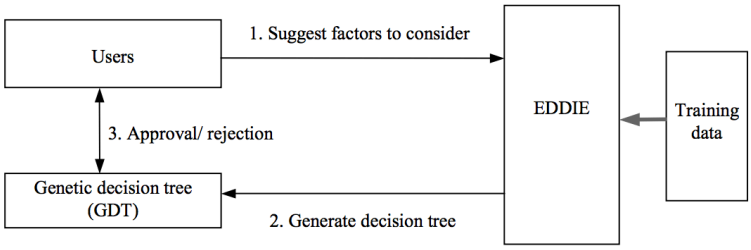


Рис. 2: Система Eddie

Конечные правила этой системы не всегда могут быть приспособлены к торговле. Система не даёт указаний к действию, лишь отвечает на вопрос: "поднимется ли цена на r % в течении n дней".

Предложенная в дипломной работе система имеет похожую архитектуру, однако правила могут быть непосредственно применены к реальной торговле.

3. Реализация

Реализация системы включает в себя применение методов, описанных выше. В частности - генерация деревьев решения посредством генетического алгоритма. Задача заключалась в нахождении значений предложенных пользователем индикаторов, которые могут сигнализировать о покупке или продаже инструмента.

Следует отметить, что аналогичные системы используют сигналы BUY/SELL, допуская при этом открытие коротких позиций и работу с ними симметрично длинным. На практике, однако, есть ограничения: адекватная оценка эффективности коротких позиций затруднена из-за необходимости оплаты заемных бумаг; срок их удержания должен быть как можно короче; временные или постоянные запреты на их открытие. Таким образом описанная в дипломной работе система работает лишь с длинными (long) позициями, и торговые указания, расположенные на терминальных узлах имеют значения Long и Close Long.

3.1. Структура дерева

Дерево решений представляет из себя связный ациклический граф, в листьях которого будут находиться решения, в данном случае это могут быть сигналы для открытия длинной позиции ($LONG$) и выхода из нее ($CLOSE\ LONG$). Процесс определения решения начинается с корня, затем в зависимости от значения узла, происходит переход либо в правый, либо в левый узел. В итоге в качестве результата выступает конечный лист, т.е. биржевой приказ купить акцию и открыть длинную позицию ($LONG$) или распродать купленный объем ($CLOSE\ LONG$). Система начинает работу с состояния $CLOSE\ LONG$, обеспечивая возможность совершения первой покупки при сигнале $LONG$. Размер позиции определяется максимально возможным количеством акций, которые можно приобрести при текущем балансе. Таким образом, весь полученный доход реинвестируется. Пример дерева представлен на рис. 3

Узел дерева представляет из себя предикат, определяющий переход к одному из двух потомков. В каждом нетерминальном узле происхо-

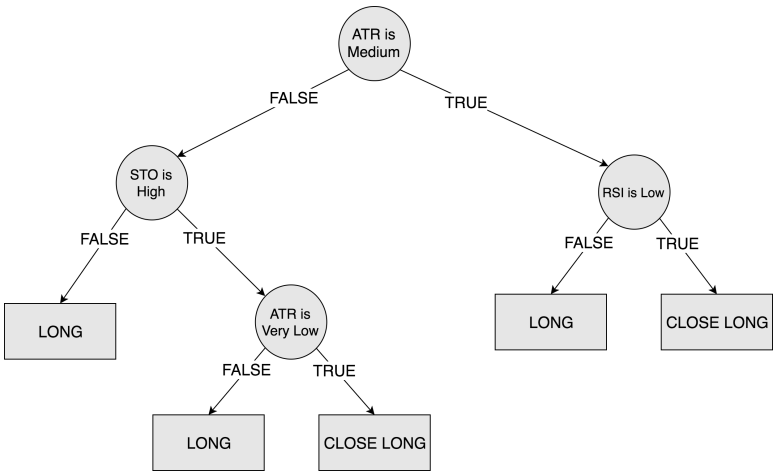


Рис. 3: Примера дерева биржевых решений

дит проверка на совпадение значения технического индикатора с определенным уровнем.

3.2. Уровень индикатора

Чаще всего индикаторы имеют вещественные значения, но их текущее состояние можно сопоставить определенному уровню, такой подход зарекомендовал себя в работе [11]. Иными словами, технический индикатор в определенный момент времени может иметь одно значение из набора: $\{$*Very Low, Low, Medium, High, Very High*$\}$.

Для вычисления этого значения, система использует n предыдущих значений индикатора, в данной системе n равен длительности периода тренировки. Затем идёт вычисление диапазонов для каждого уровня:

1. Вычисляется максимальное (max) и минимальное (min) значение индикатора за предыдущие n дней.

2. На основе этих показателей строится отрезок [min, max] и разбивается на 5 равных сегментов.

3. Текущий уровень определяется сегментом, в который попадает вещественное значение индикатора.

В случае, если значение больше max, уровню индикатора присваивается $VeryHigh$. Аналогично, при значении меньше min уровнем индикато-

ра будет $VeryLow$. В таблице 1 продемонстрирован пример результата расчёта уровней для индикатора RSI, построенного для акций IBM на конец 2007 года.

Необходимо также отметить, что текущий уровень индикатора определяется после закрытия торговой сессии и действует весь следующий торговый день.

Таблица 1: RSI диапазоны

Level	**Range**
VERY LOW	< 0.200
LOW	$[0.200; 32.562)$
MEDIUM	$[32.562; 64.924)$
HIGH	$[64.924; 97.286)$
VERY HIGH	≥ 97.286

3.3. Генетический алгоритм

Для выявления наилучшего дерева используется генетический алгоритм. Задействованы все шаги классической эволюции.

Инициализация

Начальная популяция инициализируется заданным количеством случайных деревьев, причем каждый узел создаётся путём выбора случайного индикатора и сопоставления ему случайного уровня. Высота дерева определяется количеством доступных системе индикаторов. По достижению данной высоты, генерация нетерминальных узлов прекращается и создаются случайные листья LONG/CLOSE LONG.

Отбор

С целью дальнейшего преобразования дерева генетическими операторами, отбираются наиболее приспособленные особи и тем самым формируется следующее поколение. Отбор происходит методом рулетки, у каждого дерева вероятность быть выбранным равна:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1}$$

где f_i - фитнесс-функция дерева T_i

Фитнесс-функция

Для оценки дерева был выбран коэффициент Шарпа (Sharpe Ratio) - показатель эффективности стратегии, введён W.F. Sharpe [12], определяет соотношение между прибылью и возможным риском. Вычисляется как:

$$f = \frac{r_p - R_f}{\delta(x)}, \tag{2}$$

где r_p - средний доход, R_f - доход с риском равным нулю, $\delta(x)$ - стандартное отклонение дохода. В качестве безрисковой ставки было взято значение 2.5%, как аналог 10-летних облигаций US Treasuries.

Генетические операторы

Генетические операторы построены по аналогии с работой [3] и включают операторы кроссовера и мутации.

Благодаря этим двум операциям можно достичь принципов наследственности и адаптации. Учитывая специфику древовидной структуры данных, основой для модификации служат манипуляции с терминальными и нетерминальными узлами.

В процессе кроссовера для скрещивания отбираются две особи (в виде деревьев решений). Вероятность выбора каждой согласуется с методом рулетки и расчитывается по формуле (1). Затем копии обоих деревьев обмениваются своими случайно выбранными поддеревьями и включаются в популяцию (вместе с исходными деревьями). Пример работы оператора кроссовера изображён на рис. 4.

Оператор мутации является унарным оператором, цель которого защита от преждевременной сходимости поколения (в виде множества деревьев) друг к другу. Он основан на том же принципе, что и кроссовер - в дереве выбираются случайно два узла и меняются местами. Пример работы оператора мутации изображён на рис. 5.

3.4. Удаление лишних узлов

После применения данных операторов важным шагом является устранение несогласованных узлов. На рис. 6 представлен типичный случай

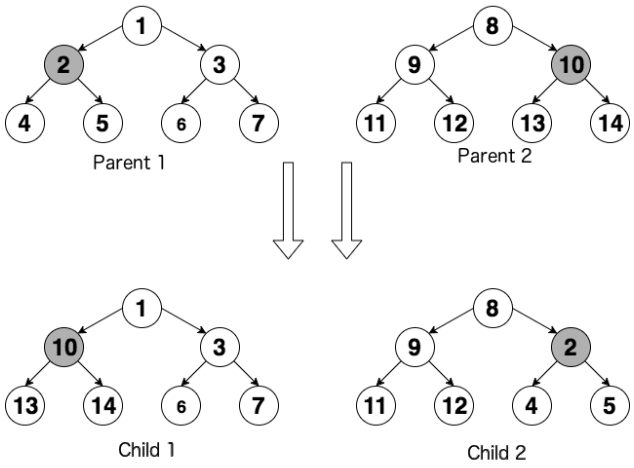


Рис. 4: Оператор кроссовера

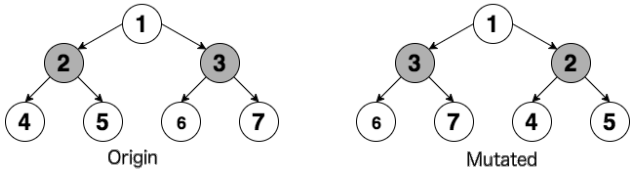


Рис. 5: Оператор мутации

подобного рода ошибок. Как можно видеть, при попадании в узел *IND1 is LOW* заранее известно, что данный предикат ложный, так как на пути от него к корню находится узел *IND1 is HIGH*. Процесс удаления лишних узлов происходит путем подъема их соответствующего потомка, в данном примере у *IND1 is HIGH* в случае истинности будет переход на *IND3 is MEDIUM* (т.е. то поддерево, куда был бы переход в случае ложности узла *IND1 is LOW*).

После генерации очередного поколения, для каждого дерева, которое было изменено путём кроссовера или мутации, запускается алгоритм удаления лишних узлов (Алгоритм 1). В итоге, дерево либо не меняет свою структуру, либо, при обнаружении лишнего предиката в узле, происходит смена узлов, согласно предложенному ниже алгоритму. На выходе получается оптимизированное дерево ($optimizedTree$), которое работает точно так же, как и изначальное.

```
Input  : replacedNode1, replacedNode2
Output: optimizedTree
// Обход потомков измененных узлов
TraversalOptimize(replacedNode1);
TraversalOptimize(replacedNode2);
Procedure TraversalOptimize(tree)
    if tree.Left ≠ Null then
        Traversal(tree.Left);
    end
    if tree.Right ≠ Null then
        Traversal(tree.Right);
    end
    FixRepetitions (tree);
/* Процедура для проверки и удаления лишних
   узлов                                        */
Procedure FixRepetitions (node)
    /* Запоминаем истинную и ложную ветку
       проверяемого узла                        */
    trueWay ⟵ node.RightTree;
    falseWay ⟵ node.LeftTree;
    currentNode ⟵ node.parent;
    /* Поднимаемся к корню и сверяем node с
       каждым узлом на пути                     */
    while currentNode ≠ Null do
        if currentNode.Name = node.Name then
            if currentNode.Value = node.Value then
                // Обнаружен идентичный предикат
                if node in currentNode.LeftTree
                 then
                    /* Если узел находится в левом
                       потомстве текущего, заменяем
                       его на ложную ветку.     */
                    node ⟵ falseWay;
                else
                    /* Иначе на истинную        */
                    node ⟵ trueWay
                end
            else
                /* Если совпадают только названия
                   индикаторов, достаточно
                   проверить, находится ли узел в
                   правом потомстве текущего. В
                   этом случае узел заведомо
                   ложный, так как отличаются
                   значения.                    */
                if node in currentNode.RightTree
                 then
                    node ⟵ falseWay;
                end
            end
        end
        /* Поднимаемся на уровень выше          */
        currentNode ⟵ currentNode.parent
    end
```

Алгоритм 1

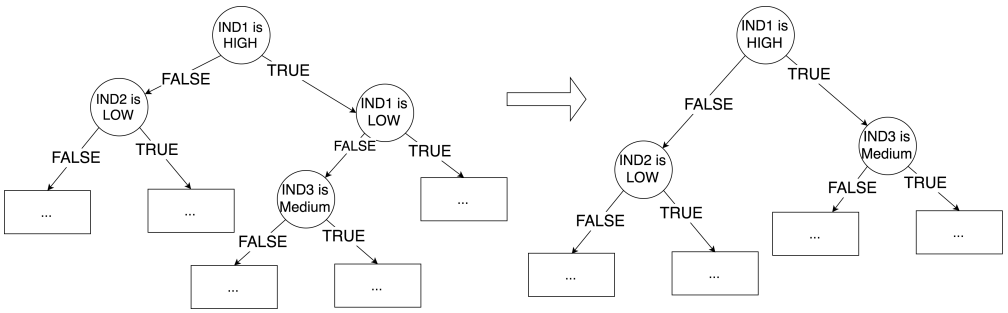


Рис. 6: Оптимизация дерева

Алгоритм принимает на вход два узла, которые меняются местами. При этом узлы могут принадлежать как разным деревьям, так и одному. Таким образом, его можно применять как в случае кроссовера, так и мутации. В данном алгоритме $replacedNode1$, $replacedNode2$ - узлы, которые поменялись местами; $RightTree$ - поддерево узла в случае истинности предиката, $LeftTree$ - поддерево в случае ложности; $Name$ - имя индикатора в предикате, $Value$ - его значение; $parent$ - родитель узла.

Оценка вычислительной сложности составляет $\mathcal{O}(n \log n)$, где n - число узлов в дереве. Такая сложность позволяет, в принципе, использовать большое количество индикаторов для построения торговой системы.

Общий алгоритм

Исходя из описанных выше особенностей, можно составить алгоритм общего вида:

- Step 1. Определение параметров генетического алгоритма: n - размер популяции, m - часть популяции для оператора мутации, l - часть популяции для кроссовера, p - количество поколений.

- Step 2. Расчет диапазонов для разбиения значений индикаторов по уровням.

- Step 3. Генерация начальной популяции. Вычисление целевой функции f_i по формуле (2), для каждой особи.
  CurrentGen $= 0$

- Step 4. Создание нового поколения. Выбор выживших методом рулетки.
  $CurrentGen = CurrentGen + 1$

- Step 5. Создание потомков. Выбор родителей методом рулетки, генерация потомков оператором кроссовера. Количество родителей должно составлять не больше $l\%$ от популяции.

- Step 6. Применение оператора мутации к случайно выбранным $m\%$ от популяции особей.

- Step 7. Вычисление целевой функции f_i по формуле (2), для каждой особи.

- Step 8. Оптимизация деревьев, полученных путём кроссовера и мутации (Алгоритм 1).

- Повторять шаги 4-8, пока $CurrentGen < p$.

3.5. Программная реализация системы

Для реализации системы был использован язык программирования Python, данный язык хорошо зарекомендовал себя в обработке больших объёмов данных.

Получение биржевых данных

С целью решения этой задачи используется открытый API портала Yahoo Finance. При помощи фреймворка *pandas*, который в свою очередь реализует данно данные скачиваются напрямую из сервера в формате csv. Затем преобразовывались в объект *pandas.DataFrame*, к которому впоследствии происходят все обращения, связанные с обработкой исторических данных. Более подробно с фреймворком *pandas* можно ознакомиться на официальном сайте [14].

Получение дневных показателей сводится к следующей строке:

```
data = web.DataReader(stock_name, 'yahoo', start, end)
```

где *stock_name* - тикер инструмента, *start* и *end* даты начала и конца требуемых данных в формате $YY - DD - MM$. Пример получаемых данных продемонстрирован на рис. 7. Важным моментом является сохранение данных непосредственно на диск, так как зачастую приходится обращаться к биржевым показателям разных диапазонов, с целью расчёта уровня того или иного индикатора.

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Adj Close
2	2012-12-31	36.799999	37.700001	36.650002	37.68	4634600	37.68
3	2012-12-28	36.91	37.400002	36.889999	36.900002	3171600	36.900002
4	2012-12-27	37.490002	37.619999	36.830002	37.299999	3422700	37.299999
5	2012-12-26	37.48	37.990002	37.259998	37.549999	2437600	37.549999
6	2012-12-24	37.759998	37.82	37.310001	37.529999	2448000	37.529999
7	2012-12-21	37.360001	38.00	37.349998	37.709999	7975900	37.709999
8	2012-12-20	38.080002	38.16	37.790001	37.869999	4330300	37.869999
9	2012-12-19	37.389999	38.25	37.349998	38.09	5853300	38.09
10	2012-12-18	37.419998	37.509998	37.18	37.490002	7652400	37.490002
11	2012-12-17	37.07	37.43	36.900002	37.419998	8928100	37.419998
12	2012-12-14	37.200001	38.099998	37.00	37.560001	17447900	37.560001
13	2012-12-13	36.009998	36.50	35.189999	35.529999	7746200	35.529999
14	2012-12-12	35.73	36.279999	35.689999	35.959999	4598200	35.959999
15	2012-12-11	35.799999	36.110001	35.470001	35.540001	5912900	35.540001
16	2012-12-10	35.450001	35.779999	35.400002	35.75	3542100	35.75
17	2012-12-07	35.400002	35.50	34.939999	35.48	2725400	35.48
18	2012-12-06	35.25	35.68	35.00	35.139999	5571900	35.139999
19	2012-12-05	35.299999	35.630001	34.860001	35.400002	4277000	35.400002

Рис. 7: Пример csv файла

Расчёт дополнительных индикаторов

Изначально в системе для расчёта индикаторов использовался фреймворк *NumPy*. Данная библиотека позволяла производить массивные векторные вычисления над объектом *pandas.DataFrame*, применяя формулу заданного индикатора. Однако с увеличением количества интересующих нас индикаторов возникла потребность в использовании готовых решений. Библиотека Ta-Lib предоставляет API для расчёта более чем 200 индикаторов, дополнительным удобством является готовая обёртка для языка Python. Для расчёта столбца значений индикатора, на вход необходимы исторические значения. Например для вычисления *Average True Range* индикатора, нужно написать:

```
talib.ATR(High, Low, Close, timeperiod=14)
```

Список всех доступных индикаторов и инструкции по установке и использованию можно найти на официальном сайте Ta-Lib [15].

Также система способна обрабатывать индикаторы полученные иным путём. Для этого необходимо и достаточно наличие столбца значений, которое затем должно слиться с общим *pyDataFrame*, это могут быть например значения из csv файла.

Реализация дерева биржевых решений

Исходя из большого количества особенностей, объект для хранения и обработки дерева решений создавался с нуля. Для этого необходимо определить терминальный и нетерминальный узел. Для нетерминального случая узел имеет ссылки на двух своих потомков, ссылку на предка и значения необходимые для определения предиката в узле (имя индикатора и его целевое значение уровня). Для листового узла достаточно информации о предке и о значении (Long, CloseLong). Необходимость хранения ссылки на предка заключается в том, что при запуске алгоритма для удаления лишних узлов требуется обход снизу вверх некоторых участков. Корневой узел имеет значение предка эквивалентное *null*. Общий вид узлов представлен на рисунке 8.

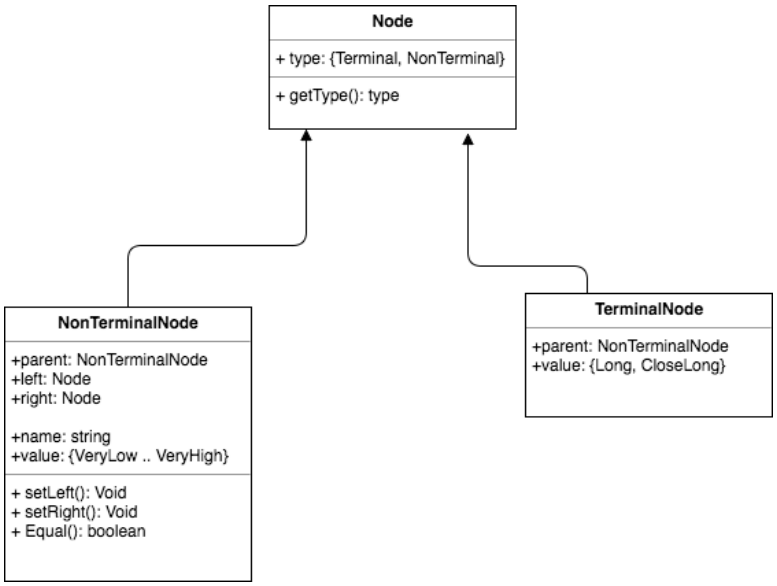


Рис. 8: Узлы дерева

Визуализация стратегий

Для визуализации деревьев решения используется инструмент ETE3. После небольшой трансформации результирующих стратегий система имеет возможность сопровождать работу графическим представлени-

ем. На рис. 9 представлен пример подобного рода изображений.

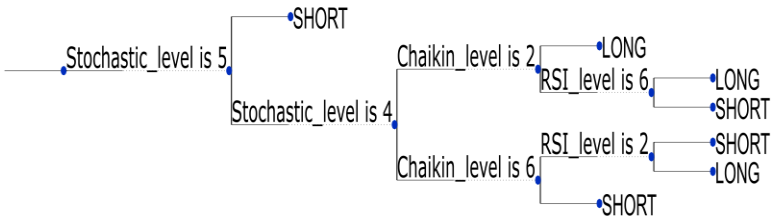


Рис. 9: Визуализация стратегии

Сбор статистики

Успешные и интересующие пользователя стратегии проходят подробное тестирование при помощи фреймворка *pyAlgoTrade*. Фитнесс-функция представляет из себя симуляцию торговли на готовых данных, в результате которой мы получаем интересующий нас коэффициент Шарпа. Однако с целью оптимизации, дополнительные метрики не вычисляются. *pyAlgoTrade* может предоставить подробную статистику по каждой из стратегий. В частности такие значения как:

1. Итоговый баланс после симуляции
2. Прибыльность стратегии
3. Коэффициент Шарпа
4. Максимальная просадка
5. Количество сделок (прибыльных, убыточных)
6. Средний доход от сделки
7. Максимальный/минимальный доход от сделки

Некоторые из этих метрик более подробно описаны в разделе 5.2.

Также с использованием инструмента *matplotlib* система иллюстрирует в отчёте графики изменения цены и целевых индикаторов. Итоговая статистика выводится в формате pdf, в Приложении А продемонстрирован пример отчёта для акций IBM.

3.6. Архитектура системы

Общая схема работы системы представлена на рис. 10.

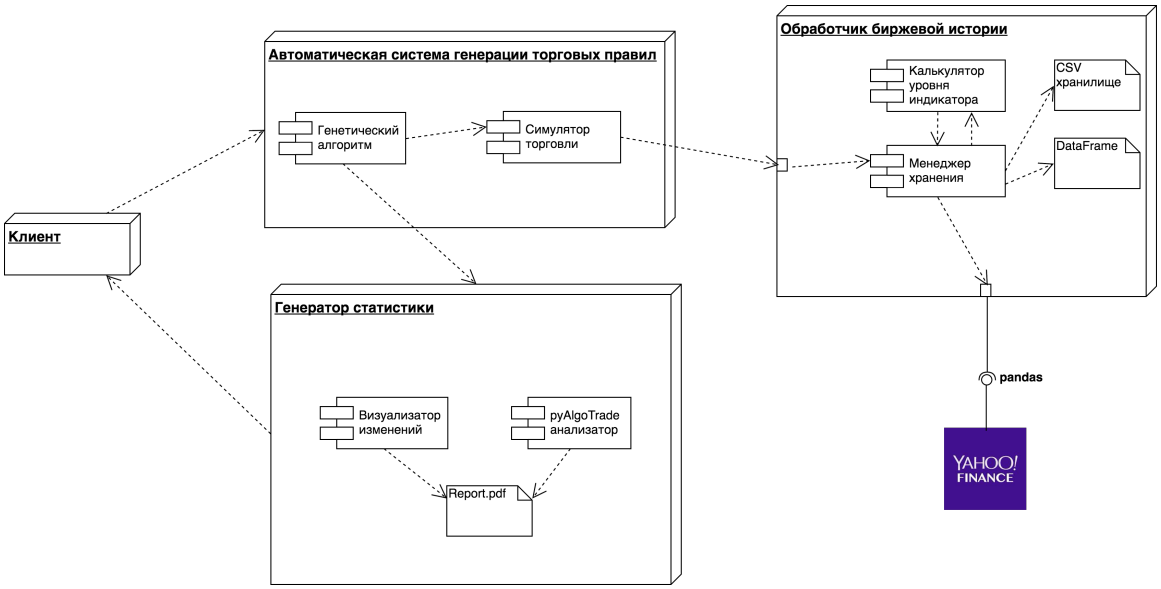


Рис. 10: Диаграмма компонентов торговой системы

Большинство из перечисленных компонентов были описаны ранее. Стоит обратить внимание на хранение исторических данных как в csv формате, так и в объектном *pandas.DataFrame*. Скачивание с сервера Yahoo происходит в формате csv файла, с целью повторного использования эти файлы сохраняются на диске и лишь по требованию конвертируются в *pandas.DataFrame*.

Модель главного бизнес-процесса изображена на рис. 11

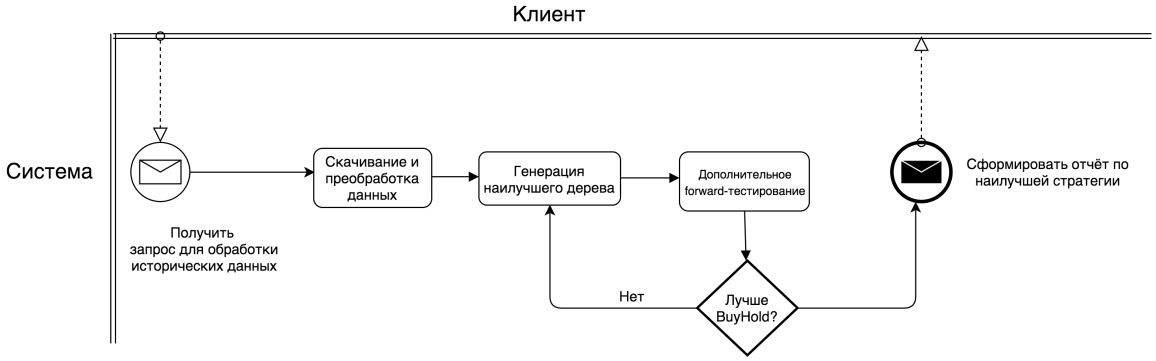


Рис. 11: BPMN диаграмма

Система получает от клиента необходимые параметры: целевой инструмент, дата начала и окончания тренировки. Генетический алгоритм

создаёт наиболее оптимальное решение на предложенных исторических данных, затем проводит дополнительное forward-тестирование (симуляцию на новых данных), в случае если полученная стратегия хуже чем *Buy and Hold* аналог, система повторно запускает генетический алгоритм, как правило подобные ситуации происходят крайне редко.

Учитывая гибкость языка *Python* большинство процессов были оформлены в виде скриптов. Однако можно составить общую аналитическую диаграмму классов, одна из вариаций представлена на рис. 12

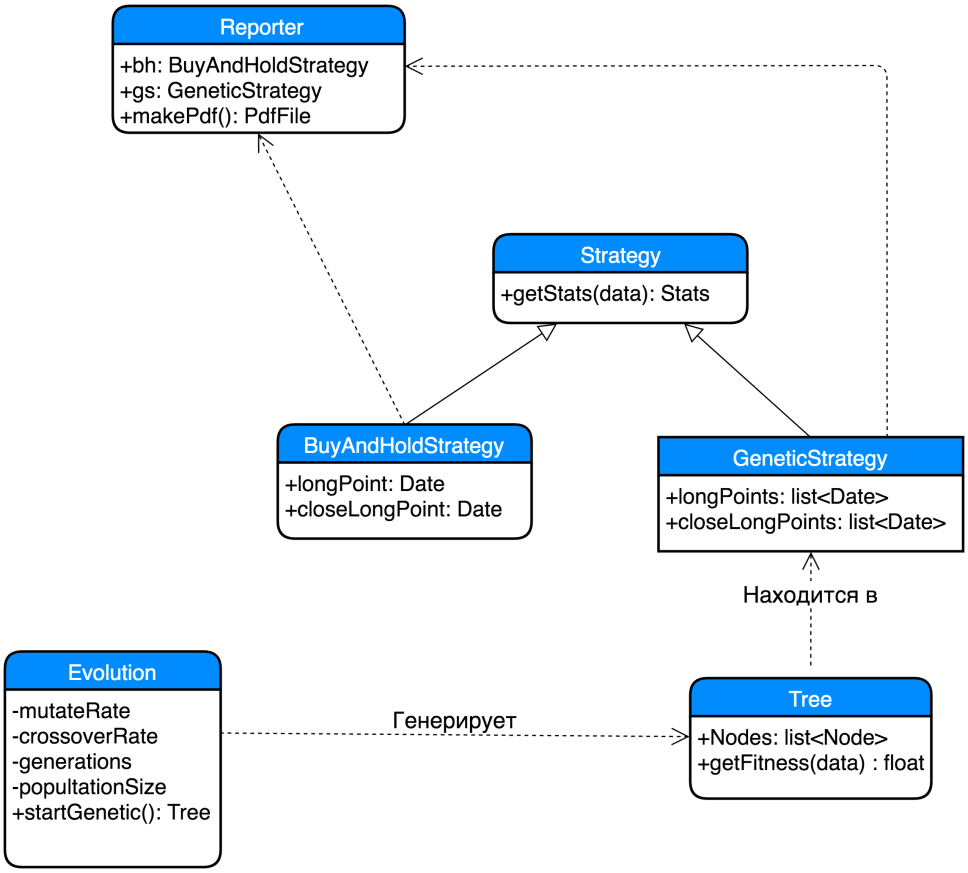


Рис. 12: Диаграмма классов

В данном случае объект *Strategy* является реализацией *pyAlgoTrade* класса, в случае с *BuyAndHoldStrategy* расширением будут служить поля указывающие на дату начала и окончания тестирования. При наступлении первой даты система купит инструмент, при наступлении второй – реализует.

GeneticStrategy имеет более сложную структуру. Используя объект *Tree*, который состоит из описанных ранее узлов, *GeneticStrategy* на-

ходит все точки на исторических данных, при которых дерево решений будет советовать покупать инструмент (*Long*), либо реализовывать его *CloseLong*. Класс *Evolution* представляет из себя генетический алгоритм. Путём диалога с пользователем скрипт получает необходимые параметры для итеративной генерации поколений, затем наилучшее дерево поступает в *Reporter* для окончательного анализа.

Как упоминалось ранее, решение, предложенное деревом решений не является окончательным. Это связано с отсутствием *Short* действий. Так например если у пользователя отсутствует объём для реализации – он не сможет исполнить *CloseLong* действие. Общий алгоритм для принятия окончательного решения представлен на рис. 13

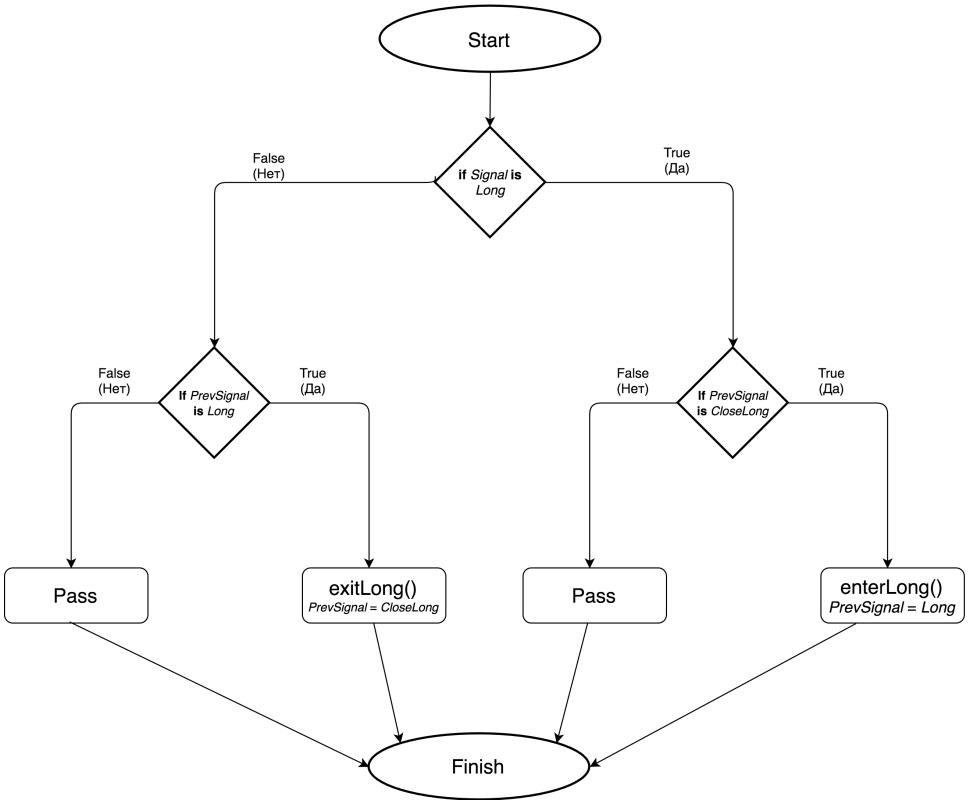


Рис. 13: Блок-схема принятия окончательного решения

4. Руководство пользователя

Код данной системы является открытым, скачать его можно по адрессу *https://github.com/iskrich/diploma*.

Также исходный код можно получить git запросом

```
git pull https://github.com/iskrich/diploma.git
```

Для запуска системы необходимо наличие языка Python версии 2.7. Для правильной работы системы должны быть установлены следующие библиотеки:

1. NumPy
2. ETE3
3. pandas
4. pyAlgoTrade

Система запускается посредством командной строки, пользователю необходимо последовательно задать три параметра: целевой инструмент, дата начала тренировки, дата окончания. Например для тренировки дерева по данным акций *Apple* за 2013 год необходимо написать:

```
python main.py "AAPL" "2013-01-03" "2013-12-30"
```

После завершения работы в папке *reports/* будет находиться итоговый отчёт по результатам работы.

5. Тестирование

5.1. Используемые индикаторы

Ниже описаны индикаторы, на базе которых проходили тесты для генерации торговых рекомендаций:

1. Relative strength index (RSI), данный осциллятор сравнивает количество подъемов цены актива за последнее время с количеством ее падений и представляет эту информацию в виде числа от 0 до 100. Вычисляется по формуле:

$$RSI = 100 - \frac{100}{1 + U/D}, \tag{3}$$

где U - количество положительных ценовых изменений, а D - количество отрицательных ценовых изменений за рассматриваемый период.

2. Stochastic oscillator (STO), показывает положение текущей цены относительно диапазона цен за определенный период, измеряется в процентах, по формуле:

$$STO_t = \frac{C_t - L_n}{H_n - L_n} \cdot 100, \tag{4}$$

где C_t - текущая цена, L_n - самая низкая цена за n периодов в прошлом, H_n - самая высокая цена за n периодов в прошлом.

3. Chaikin oscillator (CHO), помогает прогнозировать поведение индикатора накопления/распределения, который, в свою очередь, учитывает изменение цены и объёма торгов. Вначале вычисляется индикатор Accumulation/Distribution (A/D):

$$A/D_i = \frac{(Close - Low) - (High - Close)}{High - Low} \times \\ \times Volume - A/D_{i-1}, \tag{5}$$

где A/D_i — значение A/D для текущего дня,

Close — цена закрытия дня,
Low — минимальная цена дня,
High — максимальная цена дня,
Volume — объем сделок в этот день,
A/D_{i-1} — значение индикатора за предыдущий день.

Далее находим значение осциллятора:

$$CHO = EMA(A/D, 3) - EMA(A/D, 10) \tag{6}$$

где EMA(A/D, n) — n-периодная экспоненциальная скользящая средняя индикатора A/D.

5.2. Метрики

Помимо целевой функции - коэффициент Шарпа, для анализа результатов были использованы дополнительные метрики.

1. Return - представляет из себя коэффициент прибыльности в процентах и вычисляется посредством симуляции торговли на исторических данных. Чем он больше, тем лучше.

$$Return = (\frac{FinalBalance}{StartBalance} - 1) * 100, \tag{7}$$

где FinalBalance - количество денег после симуляции, а StartBalance - количество денег в начале.

2. Max drawdown - максимальное снижение величины баланса счета от своего локального максимума в процентах. Позволяет оценить максимальную потерю капитала при работе с данной стратегией. Чем он ниже, тем лучше.

3. Trades count - количество открытых позиций за рассматриваемый период. Показатель пропорционален возможной величине комиссионых издержек.

4. Buy and Hold - представляет собой доходность по акции, которую предоставляет рынок. Суть данной стратегии в покупке инструмента в самом начале и его удержание до окончания заданного торгового периода. Для сравнения были взяты показатели прибыли и коэффициент Шарпа (B&H Returns, B&H Sharpe).

5.3. Исходные данные

В качестве объекта тестирования были выбраны акции 5 крупных компаний IT сектора: Citrix Systems (CTXS), Electronic Arts (EA), eBay Inc (EBAY), Intel (INTC), Oracle (ORCL). Последовательно за каждый год происходит отбор наилучшего дерева биржевых решений и его проверка в следующем. Один год представляет из себя 251 торговый сигнал, каждый из которых соответствует определённому биржевому дню.

Следует отметить, что временные характеристики нашего алгоритма позволяют применять его к анализу 5-минутных таймфреймов и выше. Однако для ликвидных акций чем больше рассматриваемый таймфрейм, тем ниже влияние "шумовой" компоненты рынка, обусловленной непредсказуемыми действиями крупных игроков, а также случайной совокупностью действий многих биржевых роботов и трейдеров. Неустойчивость торговой системы к такой случайной компоненте приводит к преждевременным сигналам. Поэтому, аналогично работам [13] и [3], тестирование проводится на дневных исторических данных.

5.4. Аппаратные параметры системы

Серия экспериментов проводилась на машине со следующими характеристиками:

- Python 2.7
- Intel® Core™ i5 2700 Mhz
- OS X Yosemite 10.10
- 8GB (DDR3) RAM

5.5. Производительность и подбор параметров

Время работы системы при различных размерах популяции представлено в таблице 2, система работала с данными размером в один биржевой год. В таблице приведены усредненные показатели тестов, запущенных пять раз подряд с целью устранения влияния системного кэша на производительность.

Таблица 2: Working time

Размер	**Время (с)**
10	14.723
50	54.260
100	116.564
200	183.221

Наиболее трудозатратные операции - алгоритм удаления лишних узлов (Алгоритм 1) и вычисление уровней индикатора. В измерения не включены временные затраты на загрузку ценовых данных и подсчет статистики.

Таким образом для запуска экспериментов были использованы следующие параметры:

Таблица 3: Параметры генетического алгоритма

Вероятность кроссовера	35%
Вероятность мутации	5%
Количество поколений	25
Размер популяции	200
Количество тех. индикаторов	3

Все параметры подобраны эмпирическим путем для обеспечения баланса качества и времени выполнения. Иногда от генетического алгоритма может потребоваться устойчивость результата, то есть при последовательных запусках системы с одинаковыми параметрами должно выдаваться одинаковое дерево решений. В случае трех технических индикаторов для этого потребуется повысить кол-во особей до 800 и поколений до 50.

5.6. Результаты тестирования

Результаты представлены в Таблицах 4 - 8. Строчки с лучшими показателями системы, чем у стратегии B&H, выделены серым цветом.

Очевидно, что эффективность построенных торговых стратегий существенно зависит от применяемого набора технических индикаторов и их параметров. В связи с этим в будущем необходимо исследовать применимость других технических индикаторов и автоматизировать выбор оптимальных параметров индикаторов.

Таблица 4: CTXS результаты

Year	Sharpe	B&H Sharpe	Return	B&H returns	Max drawdown	Trades Count
2007 Train	2.73	1.26	59.64%	36.16%	6.92%	42
2008 Test	-0.82	-0.73	-20.23%	-35.72%	36.66%	28
2008 Train	1.60	-0.73	26.88%	-35.72%	6.70%	26
2009 Test	-0.05	1.52	1.90%	67.20%	6.57%	16
2009 Train	3.42	1.52	102.75%	67.20%	7.96%	40
2010 Test	1.64	1.34	19.64%	54.37%	4.08%	18
2010 Train	1.82	1.34	64.59%	54.37%	10.23%	15
2011 Test	0.29	-0.07	6.66%	-9.04%	28.50%	14
2011 Train	1.53	-0.07	40.58%	-9.04%	8.58%	14
2012 Test	0.65	0.24	11.06%	5.26%	9.25%	21
2012 Train	1.57	0.24	12.60%	5.26%	2.76%	15
2013 Test	-0.78	-0.32	-3.27%	-8.77%	9.68%	18
2013 Train	2.67	-0.32	37.12%	-8.77%	3.56%	30
2014 Test	1.85	0.16	21.80%	3.66%	2.99%	34
2014 Train	3.84	0.16	46.11%	3.66%	1.51%	37
2015 Test	3.00	0.69	38.48%	19.96%	4.37%	34
2015 Train	1.69	0.69	18.37%	19.96%	1.82%	18
2016 Test	1.16	0.73	10.98%	18.66%	3.88%	35

Таблица 5: EA результаты

Year	Sharpe	B&H Sharpe	Return	B&H returns	Max drawdown	Trades Count
2007 Train	2.74	0.45	54.25%	11.56%	9.78%	70
2008 Test	-1.30	-1.94	-33.88%	-65.56%	44.81%	38
2008 Train	3.32	-1.94	132.34%	-65.56%	7.19%	54
2009 Test	1.87	0.19	28.07%	1.50%	4.35%	30
2009 Train	2.72	0.19	71.93%	1.50%	8.78%	29
2010 Test	0.54	-0.25	10.76%	-8.73%	9.25%	22
2010 Train	2.78	-0.25	43.72%	-8.73%	4.86%	52
2011 Test	-0.05	0.67	0.11%	23.08%	13.33%	34
2011 Train	1.32	0.67	46.63%	23.08%	20.28%	14
2012 Test	-0.58	-0.95	-16.38%	-27.89%	33.46%	12
2012 Train	1.46	-0.95	24.23%	-27.89%	7.94%	39
2013 Test	1.38	1.30	24.36%	53.26%	4.34%	44
2013 Train	3.62	1.30	58.90%	53.26%	5.01%	57
2014 Test	2.53	2.22	26.16%	97.53%	3.33%	46
2014 Train	5.10	2.22	112.49%	97.53%	4.00%	36
2015 Test	1.46	1.33	32.30%	44.03%	11.97%	41
2015 Train	4.59	1.33	95.09%	44.03%	4.00%	74
2016 Test	0.44	0.71	8.71%	19.65%	12.76%	52

Таблица 6: EBAY результаты

Year	Sharpe	B&H Sharpe	Return	B&H returns	Max drawdown	Trades Count
2007 Train	1.90	0.36	47.86%	9.01%	9.78%	39
2008 Test	-0.77	-1.42	-26.94%	-52.55%	40.55%	24
2008 Train	2.70	-1.42	74.23%	-52.55%	6.64%	33
2009 Test	3.38	1.24	123.44%	55.25%	8.52%	37
2009 Train	2.92	1.24	78.65%	55.25%	8.14%	36
2010 Test	0.83	0.62	11.11%	16.91%	6.39%	12
2010 Train	1.49	0.62	24.08%	16.91%	6.39%	27
2011 Test	0.85	0.24	12.15%	4.68%	7.84%	7
2011 Train	1.45	0.24	25.98%	4.68%	8.23%	30
2012 Test	1.72	1.73	21.82%	58.61%	4.96%	20
2012 Train	2.23	1.73	55.22%	58.61%	7.50%	55
2013 Test	-1.57	0.07	-21.31%	1.61%	27.20%	31
2013 Train	1.36	0.07	15.80%	1.61%	2.86%	39
2014 Test	0.68	0.20	8.71%	4.62%	6.12%	37
2014 Train	3.20	0.20	28.06%	4.62%	3.05%	28
2015 Test	-0.38	-0.69	-28.73%	-45.02%	48.15%	25
2015 Train	3.71	-0.69	55.73%	-45.02%	2.98%	48
2016 Test	0.14	0.45	3.32%	11.61%	18.19%	16

Таблица 7: INTC результаты

Year	Sharpe	B&H Sharpe	Return	B&H returns	Max drawdown	Trades Count
2007 Train	1.49	1.00	35.18%	26.54%	11.73%	36
2008 Test	-0.57	-0.93	-24.27%	-39.51%	38.93%	19
2008 Train	3.28	-0.93	68.39%	-39.51%	2.63%	45
2009 Test	1.35	0.90	18.79%	30.67%	5.21%	49
2009 Train	4.09	0.90	94.73%	30.67%	3.22%	36
2010 Test	-1.11	0.01	-7.79%	0.04%	13.79%	11
2010 Train	1.55	0.01	20.00%	0.04%	4.01%	12
2011 Test	0.66	0.55	6.92%	13.98%	1.16%	7
2011 Train	1.23	0.55	24.10%	13.98%	9.28%	41
2012 Test	0.62	-0.86	8.90%	-14.49%	8.15%	30
2012 Train	0.64	-0.86	7.47%	-14.49%	6.22%	47
2013 Test	0.48	0.85	6.76%	17.98%	6.00%	30
2013 Train	2.73	0.85	31.64%	17.98%	4.69%	30
2014 Test	3.53	1.59	51.25%	39.15%	4.03%	38
2014 Train	3.53	1.59	51.25%	39.15%	4.03%	38
2015 Test	3.11	-0.10	45.35%	-2.03%	2.43%	42
2015 Train	3.11	-0.10	45.35%	-2.03%	2.43%	42
2016 Test	-0.35	0.30	-0.91%	6.75%	7.26%	17

Таблица 8: ORCL результаты

Year	Sharpe	B&H Sharpe	Return	B&H returns	Max drawdown	Trades Count
2007 Train	2.38	0.96	35.94%	26.42%	4.23%	48
2008 Test	0.12	-0.38	2.86%	-20.45%	15.15%	42
2008 Train	2.74	-0.38	46.03%	-20.45%	6.35%	43
2009 Test	4.25	1.04	94.30%	32.90%	2.49%	46
2009 Train	4.30	1.04	118.95%	32.90%	3.29%	68
2010 Test	1.82	0.99	29.85%	24.37%	5.00%	58
2010 Train	1.58	0.99	21.62%	24.37%	3.68%	34
2011 Test	-0.27	-0.50	0.40%	-16.88%	6.75%	12
2011 Train	1.30	-0.50	22.14%	-16.88%	14.05%	40
2012 Test	0.73	1.19	8.22%	25.65%	4.98%	35
2012 Train	1.94	1.19	34.01%	25.65%	8.55%	32
2013 Test	-0.38	0.39	-5.24%	8.61%	17.23%	30
2013 Train	4.10	0.39	35.41%	8.61%	1.92%	43
2014 Test	3.89	0.89	34.14%	18.79%	1.98%	39
2014 Train	4.22	0.89	42.84%	18.79%	2.19%	49
2015 Test	2.19	-0.85	25.82%	-14.37%	5.39%	42
2015 Train	1.97	-0.85	15.50%	-14.37%	2.16%	29
2016 Test	-1.48	0.33	-8.11%	6.92%	9.41%	16

Как можно видеть на представленных примерах, в среднем в 6 из 9 случаев интегральный коэффициент Шарпа оказался лучше у нашей системы, чем у рынка. При этом максимальная потеря капитала составила 48% против 65% при простом инвестировании. Это означает, что существуют акции и периоды для которых данный подход может быть эффективным. Количество входов в позицию меняется от 58 до 7, что свидетельствует о невысокой чувствительности получаемых правил. Это может быть обусловлено тем, что в качестве скользящего интервала для определения диапазонов уровней был взят 1 год. Уменьшение этого интервала ведет к увеличению скорости реакции системы, но возникает большее количество ложных сигналов.

Отметим, что в процессе тренировки данная торговая система, настраивается на определенное соотношение трендовых и боковых движений курса акции. Если в следующем периоде это соотношение меняется, то трейдер может ожидать как прибыль, так и убыток. Данный факт присущ большинству алгоритмических систем, построенных на методах технического анализа.

Заключение

Таким образом, в ходе данной работы была создана автоматическая система генерации торговых рекомендаций. На основе исторических данных генетический алгоритм находит паттерны для сигналов покупки и продажи. Преимуществом данной системы является динамичный расчёт уровня исходных индикаторов. Архитектура системы позволяет работать практически с любым набором технических индикаторов.

По окончанию программной реализации был разработан комплекс, который помимо оптимального биржевого дерева решений также производил статистику и формировал полноценные отчёты об итоговой стратегии.

Данная система была представлена в статусе курсовой работы на конференциях:

1. СПИСОК-2016 [17]
2. Региональная информатика (РИ-2016) [18]

Обновлённая версия системы в 2017 году была показана на:

1. The 20th conference of the Open Innovations Association FRUCT [19]
2. Second Conference on Software Engineering and Information Management SEIM'17 [20]

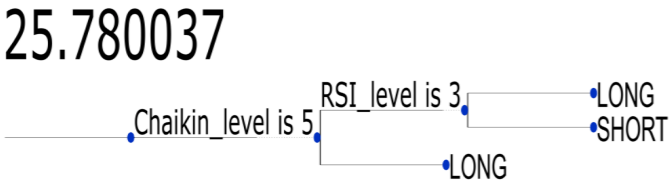
Приложение А: Пример отчёта системы

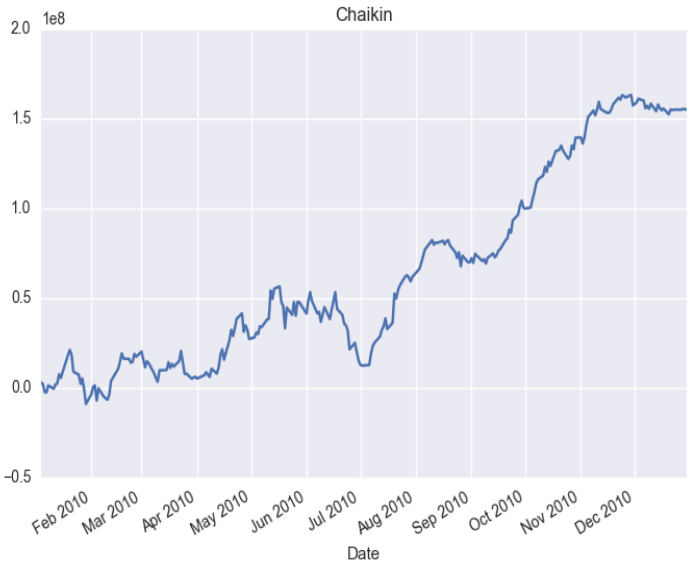
Информация о стратегии.

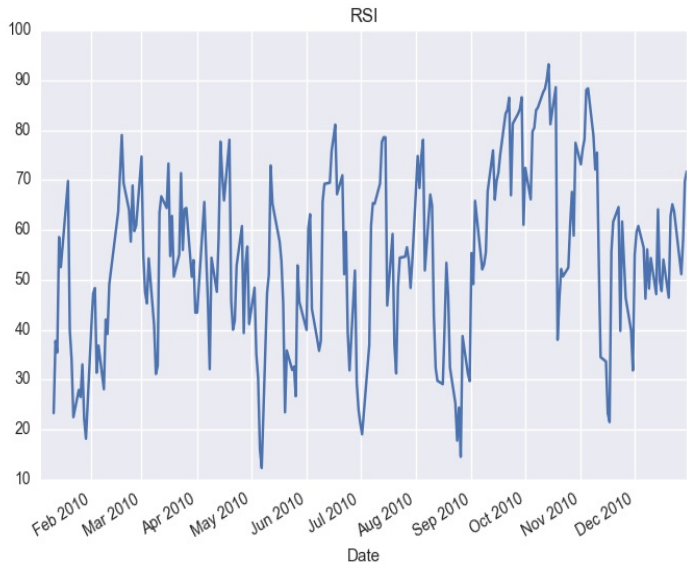
Инструмент: IBM
График изменения цены для данного инструмента

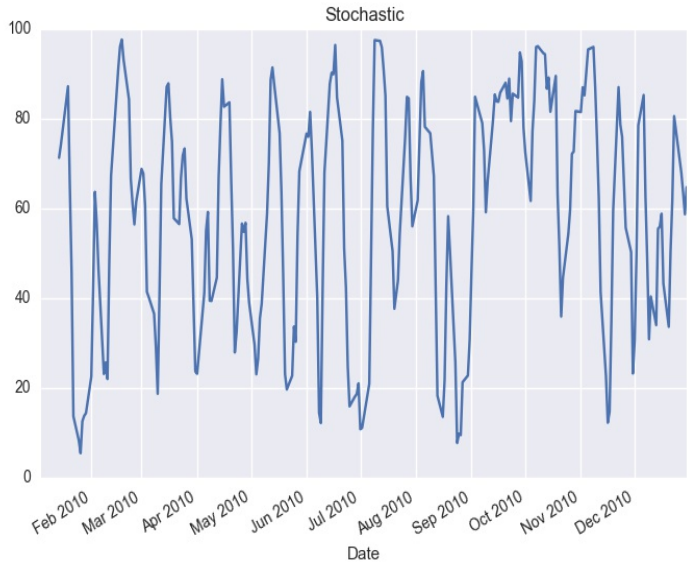
Период для тестирования 2010-01-03 по 2010-12-30

Полученная стратегия:









Результаты для периода 2011-01-03 по 2011-12-30:

Параметр	Значение
`Final portfolio value`	1160245.92685

Параметр	Значение
Cumulative returns	16.0245926851
Sharpe ratio	0.623705770782
Max. drawdown	10.9913037851
Total trades	26
Avg. profit	6698.06383946
Profits std. dev	0.0
Max. profit	99706.655592
Min. profit	-81957.12987
Profitable trades	15
Unprofitable trades	11

Список литературы

[1] Giacomel F., Galante R., Pereira A. An Algorithmic Trading Agent based on a Neural Network Ensemble: a Case of Study in North American and Brazilian Stock Markets //Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on. – IEEE, 2015. – Т. 2. – С. 230-233.

[2] Schoreels C., Logan B., Garibaldi J. M. Agent based genetic algorithm employing financial technical analysis for making trading decisions using historical equity market data //Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on. – IEEE, 2004. – С. 421-424.

[3] Potvin, Jean-Yves, Patrick Soriano, and Maxime Vallée. "Generating trading rules on the stock markets with genetic programming." Computers & Operations Research 31.7 (2004): 1033-1047.

[4] Laurent H., RIVEST R. L. Construction optimal binary decision trees is NP-complete //Information Processing Letters. – 1976.

[5] Ochotorena C. N. et al. Robust stock trading using fuzzy decision trees //Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on. – IEEE, 2012. – С. 1-8.

[6] Iba H., Aranha C. C. Practical Applications of Evolutionary Computation to Financial Engineering. – Springer, 2012.

[7] Rakhmawati N. A., Suryani E., ITS J. R. Decision for Buying and Selling Stock with Decision Tree Algorithm. – 2006.

[8] Al-Radaideh Q. A., Assaf A., Alnagi E. Predicting stock prices using data mining techniques //The International Arab Conference on Information Technology (ACIT'2013), 2013.

[9] Carvalho D. R., Freitas A. A. A hybrid decision tree/genetic algorithm

method for data mining //Information Sciences. – 2004. – Т. 163. – №. 1. – С. 13-35.

[10] Tsang E. P. K., Li J., Butler J. M. EDDIE beats the bookies //Softw., Pract. Exper. – 1998. – Т. 28. – №. 10. – С. 1033-1043.

[11] Ghandar A. et al. Computational intelligence for evolving trading rules //IEEE Transactions on Evolutionary Computation. – 2009. – Т. 13. – №. 1. – С. 71-86.

[12] Sharpe W. F. The sharpe ratio //The journal of portfolio management. – 1994. – Т. 21. – №. 1. – С. 49-58.

[13] Nugroho F. X. S. D., Adji T. B., Fauziati S. Decision support system for stock trading using multiple indicators decision tree //Information Technology, Computer and Electrical Engineering (ICITACEE), 2014 1st International Conference on. – IEEE, 2014. – С. 291-296.

[14] Python Data Analysis Library [Электронный ресурс]. – http://pandas.pydata.org/

[15] TA-Lib : Technical Analysis Library [Электронный ресурс]. – http://www.ta-lib.org/function.html

[16] ETE Toolkit [Электронный ресурс]. – http://etetoolkit.org/

[17] Искрич Д., Григорьев Д., Генерация правил торговой системы с помощью анализа исторических данных генетическим алгоритмом //СПИСОК 2016: Материалы 6-й всероссийской научной конференции по проблемам информатики. - 2016 - С. 81-87.

[18] Искрич Д., Григорьев Д., Применение деревьев решений в области биржевого трейдинга" //Региональная информатика «РИ-2016» - 2016 - С. 517

[19] Iskrich D., Grigoriev D. Generating Long-Term Trading System Rules Using a Genetic Algorithm Based on Analyzing Historical Data,

//Proceedings of the International FRUCT Conference on Intelligence, Social Media and Web - 2017.

[20] Iskrich D, Grigoriev D. Automatic Creation of Stock Trading Rules on the Basis of Decision Trees //Second Conference on Software Engineering and Information Management SEIM'17, 2017.